Peter Dillinger

Implications by Induction

Proving by induction conjectures that are implications is especially tricky. We will generate a kind of roadmap for such proofs. Given definitional axioms

```
(integer-listp x)
=
(if (endp x)
  (equal x nil)
  (and (integerp (car x))
       (integer-listp (cdr x))))
```

```
(app x y)
=
(if (endp x)
  y
  (cons (car x)
        (app (cdr x) y)))
```

Let us prove

```
(implies (and (integer-listp x)
              (integer-listp y))
         (integer-listp (app x y)))
```

This certainly requires induction because `integer-listp` would have to walk over all the elements of `x` to arrive at the assumption `(integer-listp y)`. Also, we want to induct on the variable `x`. The scheme we want is based on `(integer-listp x)`. In fact, this is the same as the induction scheme based on `(true-listp x)`, because the test for the base case and the parameters to the recursive call are the same. Here's the scheme:

```
(implies (endp x)
         φ)
```

```
(implies (and (not (endp x))
              (let ((x (cdr x)))
                φ))
         φ)
```

And since φ is that big implication, the second proof obligation, the inductive step, is particularly big and nasty. (By the way, `(let ((x (cdr x))) φ)` is called the **induction hypothesis**.) It's probably not even clear how to complete such a proof. So let's consider such proofs in the abstract and figure out how to prove them.

Suppose we are trying to prove something of the form

```
(implies H C)
```

by induction. `H` is for “hypothesis” and `C` is for “conclusion.” And below we will use `B` for the base test, `H'` for `H` with variables replaced for the induction hypothesis, and `C'` for `C` with variables replaced for the induction hypothesis. To prove `(implies H C)` by induction, we need to prove

```
(implies B                                   Base Case (old)
         (implies H C))

(implies (and (not B)                        Induction Step (old)
              (implies H' C'))
         (implies H C))
```

Let's start with the base case. Because ((B /\ H) → C) → (B → (H → C)) is a boolean tautology, we can instead prove

```
(implies (and B H)                           Modified Base Case
         C)
```

for the base case. Thus, we can use `B` and `H` as assumptions in proving `C`.

To simplify the induction step, we first make a similar change:

```
(implies (and (not B)                        Induction Step (not final)
              H
              (implies H' C'))
         C)
```

Typically, we want to use `C'` in proving `C`. (For example, in our example, `C` is `(integer-listp (app x y))` and `C'` is `(integer-listp (app (cdr x) y))`.) But in order to use C', we need to know that H' is true. We can assume (not B) and H in proving H', and this is another proof obligation:

```
(implies (and (not B)                        Induction Hypothesis Chaining
              H)
         H')
```

Assuming we have proven this, that simplifies the induction step by two applications of modus ponens, and we have the final form of the induction step:

```
(implies (and (not B)                        Modified Induction Step
              H
              H'
              C'))
         C)
```

So instead of some big formulas, we have a roadmap of what to prove with the Modified Base Case, Induction Hypothesis Chaining, and Modified Induction Step. Let's see how this works for our example.

```
(implies (and (integer-listp x)
              (integer-listp y))
         (integer-listp (app x y)))
```

In this case, using induction based on `(integer-listp x)`, we have

```
B = (endp x)

H = (and (integer-listp x)
         (integer-listp y))

H'= (and (integer-listp (cdr x))
         (integer-listp y))

C = (integer-listp (app x y))

C'= (integer-listp (app (cdr x) y))
```

The modified base case is then (collapsing ANDs)

```
(implies (and (endp x)
              (integer-listp x)
              (integer-listp y))
         (integer-listp (app x y)))
```

The induction hypothesis chaining is (collapsing ANDs again)

```
(implies (and (not (endp x))
              (integer-listp x)
              (integer-listp y))
         (and (integer-listp (cdr x))
              (integer-listp y)))
```

and the modified induction step is (collapsing ANDs and redundant hypothesis)

```
(implies (and (not (endp x))
              (integer-listp x)
              (integer-listp y)
              (integer-listp (cdr x))
              (integer-listp (app (cdr x) y)))
         (integer-listp (app x y)))
```